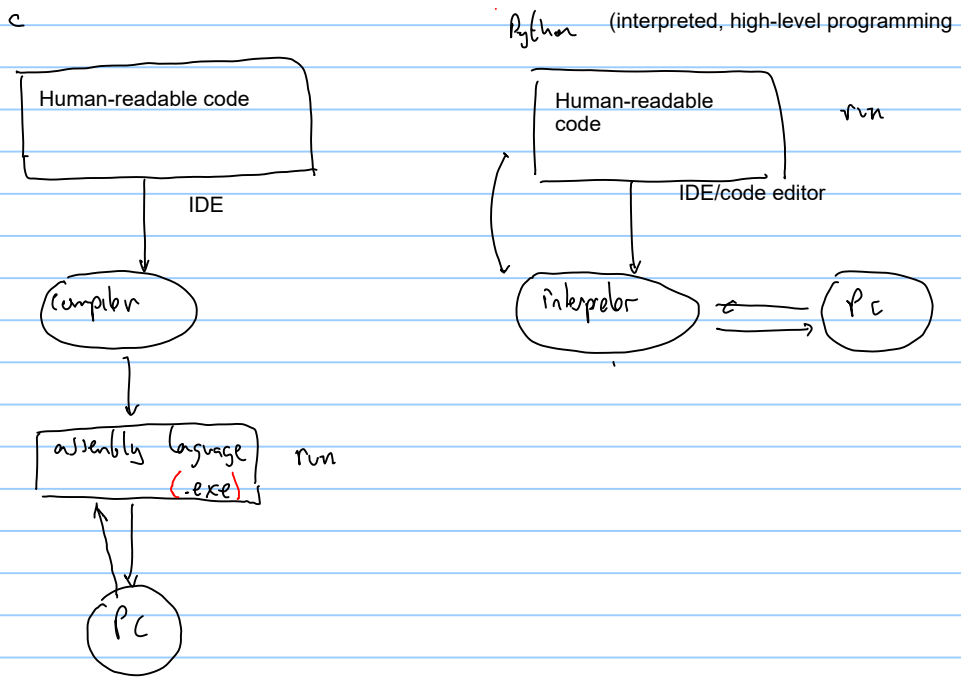
# Notes for lecture 20210302

Note Title 02.03.2021

Differences on how Python and "C" run




Python is open-source, free, with a lot of libraries (toolboxes) and there is a lot of help resources on the web. The sintax is somewhat similar to Matlab (e.g. variables do not need to be declared beforehand).
You can run Python on your computer by:
-Installing directly from https://www.python.org/ and then installing the libraries needed. Libraries are installed by calling Python from the command line. Once Python opens, use the command: "pip install name_of_package". You can also install a specific version of the package, if needed.
-Alternatively, you can download a "package" with Python and a lot of libraries, like anaconda (from https://anaconda.org/). In this case, to install additional libraries you must use anaconda's command line or interface.

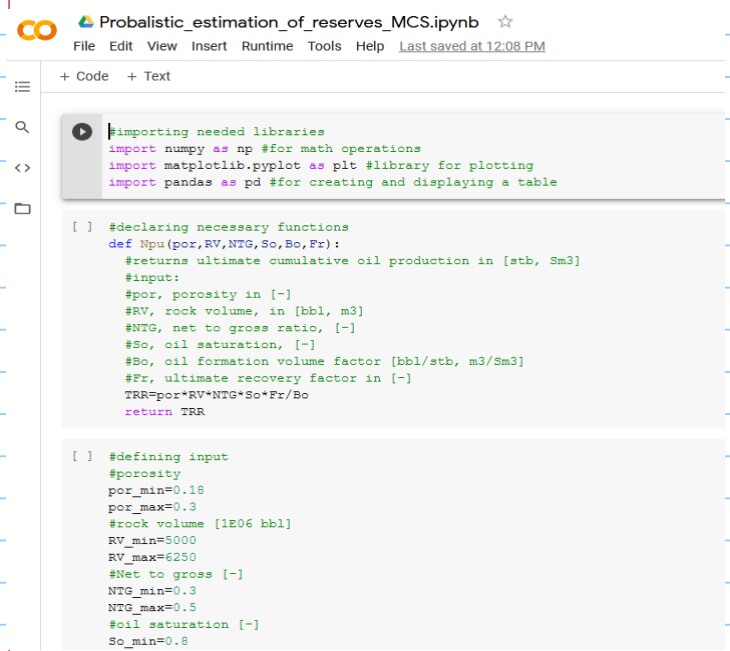
Some popular libraries are numpy and scipy (for math and scientific computing), matplotlib (for plotting), pandas for statistics, etc.

Typically, one uses a code editor or an IDE (Integrated development Environment) to write, debug and test code. For this, you typically need to:
-Install the programming language on your computer and libraries (if needed)
-Configure the editor to work with the programming language.

When you want to run parts of the code, you have to use the "debugging" mode that can allow you to run line by line or jump between sections

Some popular IDEs for Python are Pycharm and Visual studio code.

jupyter notebook is sort of a code editor for Python. It is an application that uses your browser. It allows to run interactively blocks of code. You can also add annotations, images, etc, so it can work as a interactive document. It is mainly used for solving tasks interactively, and to "prototype" code.



You can install Jupyter notebook from their website (https://jupyter.org/), it is included on the Anaconda package. There are also some websites that allow you to run Jupyter notebooks in their servers. An example is google collaboratory (https://colab.research.google.com)

```
#importing needed libraries
import numpy as np #for math operations
import matplotlib.pyplot as plt #library for plotting
import pandas as pd #for creating and displaying a table
```

```
[3] #declaring necessary functions
def Npu(por,RV,NTG,So,Bo,Fr):
  #returns ultimate cumulative oil production in [stb, Sm3]
  #input:
  #por, porosity in [-]
  #RV, rock volume, in [bbl, m3]
  #NTG, net to gross ratio, [-]
  #So, oil saturation, [-]
  #Bo, oil formation volume factor [bbl/stb, m3/Sm3]
  #Fr, ultimate recovery factor in [-]
  TRR=por*RV*NTG*So*Fr/Bo
  return TRR
```

```
[6] #defining input
#porosity
por_min=0.18
por_max=0.3
#rock volume [1E06 bbl]
RV_min=5000
RV_max=6250
#Net to gross [-]
NTG_min=0.3
NTG_max=0.5
#oil saturation [-]
So_min=0.8
So_max=0.9
#Oil formation volume factor [bbl/stb]
Bo_min=1.35
Bo_max=1.6
#recovery factor, Fr, [-]
Fr_min=0.18
Fr_max=0.35
Fr_mode=0.25
```

| | Rock volume | Porosity | Net to Gros N/G | Oil Saturation So=(1-Sw) | Formation Volume Bo | Ultimate Recovery Factor Fr |
|---|---|---|---|---|---|---|
| | bbl | fraction | fraction | fraction | Res bbl/STB | fraction |
| Min | 2000000000 | 0.18 | 0.3 | 0.8 | 1.35 | 0.42 |
| Max | 2500000000 | 0.3 | 0.5 | 0.9 | 1.6 | 0.65 |

```
[9] #creating random samples
n=1000  #number of samples
por=np.random.uniform(por_min,por_max,n)
RV=np.random.uniform(RV_min,RV_max,n)
NTG=np.random.uniform(NTG_min,NTG_max,n)
So=np.random.uniform(So_min,So_max,n)
Bo=np.random.uniform(Bo_min,Bo_max,n)
Fr=np.random.triangular(Fr_min,Fr_mode,Fr_max,n)
```

equivalent to

```
Function x_uniform(a, b)
    'value of the variable x for a uniform distribution
    'a is the minimum value of x
    'b is the maximum value of x
    'U is the the random number
    Application.Volatile (True)
    U = Rnd()
    x_uniform = a + (b - a) * U
End Function
```

equivalent to

```
Function x_Triangular(a, b, c)
    'value of the variable x for a Triangular distribution
    'a is the minimum value of x
    'b is the maximum value of x
    'c is the mode value of x
    'U is the the random number
    Application.Volatile (True)
    U = Rnd()
    F_c = (c - a) / (b - a)
      If F_c > U Then
        x_Triangular = a + Sqr((b - a) * (c - a) * U)
      Else
        x_Triangular = b - Sqr((b - a) * (b - c) * (1 - U))
      End If
End Function
```

```
[13] #MC simulation
TRR=Npu(por,RV,NTG,So,Bo,Fr)
```

| | Rock volume | Porosity | Net to Gros N/G | Oil Saturation So=(1-Sw) | Formation Volume Bo | Ultimate Recovery Factor Fr | |
|---|---|---|---|---|---|---|---|
| | bbl | fraction | fraction | fraction | Res bbl/STB | fraction | |
| Min | 2000000000 | 0.18 | 0.3 | 0.8 | 1.35 | 0.42 | |
| Max | 2500000000 | 0.3 | 0.5 | 0.9 | 1.6 | 0.65 | |
| | | | | | | | |
| MC it [-] | Rock volume bbl | Porosity fraction | N/G fraction | So=(1-Sw) fraction | Bo Res bbl/STB | Fr fraction | Npu [stb] |
| 1 | 2012066161 | 0.18926 | 0.4099759 | 0.88329645 | 1.42513865 | 0.425175962 | 4.11E+07 |
| 2 | 2102438113 | 0.26087 | 0.4857407 | 0.807186328 | 1.354060909 | 0.611151706 | 9.71E+07 |
| 3 | 2227585607 | 0.23525 | 0.4005316 | 0.850604772 | 1.580405627 | 0.527296311 | 5.96E+07 |
| 4 | 2141227076 | 0.28532 | 0.3038112 | 0.877732837 | 1.477701936 | 0.514140587 | 5.67E+07 |
| 5 | 2200610567 | 0.29046 | 0.3220469 | 0.829377484 | 1.365998187 | 0.584485561 | 7.31E+07 |

```
[18] #frequency analysis on the results
     nr_bins=15
     bins=np.linspace(TRR.min(),TRR.max(),nr_bins)
     counts,bins=np.histogram(TRR,bins=bins)
     pdf=counts/n
     cdf=np.cumsum(pdf)
```

the function np.histogram works in the following way

- min 0

< 1 — Counted as part of the 2nd bin (including min, excluding 1) 0<=X<1

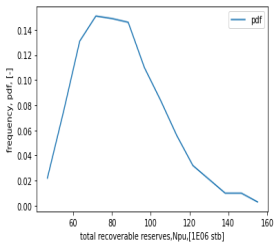
< 2 — Counted as part of the 3rd bin (including 1, excluding 2) 1<=X<2

- 3

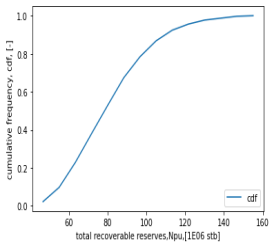
**This is why**

Number of elements in pdf vector = Number of elements in the bin vector - 1

```
[21] #plot pdf
     plt.xlabel("total recoverable reserves,Npu,[1E06 stb]")
     plt.ylabel('frequency, pdf, [-]')
     plt.plot(bins[1:],pdf,label='pdf')
     plt.legend(loc="upper right")
     plt.show()
```




```
[22] #plot cdf
     plt.xlabel("total recoverable reserves,Npu,[1E06 stb]")
     plt.ylabel('cumulative frequency, cdf, [-]')
     plt.plot(bins[1:],cdf,label='cdf')
     plt.legend(loc="lower right")
     plt.show()
```




```
[25] #create a summary table
     summary_table=pd.DataFrame({
         "Variable":['Mean [1E06 stb]','Mode [1E06 stb]','Minimum [1E06 stb]','Maximum [1E06 stb]','P90 [1E06 stb]','P50 [1E06 stb','P10 [1E06 stb]'],
         "Value":[TRR.mean(),83,TRR.min(),TRR.max(),np.percentile(TRR,10),np.percentile(TRR,50),np.percentile(TRR,90)]
     })
     summary_table
```

| | Variable | Value |
|---|---|---|
| 0 | Mean [1E06 stb] | 80.808664 |
| 1 | Mode [1E06 stb] | 83.000000 |
| 2 | Minimum [1E06 stb] | 38.385094 |
| 3 | Maximum [1E06 stb] | 155.068618 |
| 4 | P90 [1E06 stb] | 55.492273 |

BE CAREFUL
In petroleum engineering, P90 often refers to the the 90th percentile of the inverse cdf. Therefore, to calculate P90 from the regular cdf, you have to use the 10th percentile.